UNCLASSIFIED

Unique Identification of Data for Global Naming Across Arbitrary Data Systems*

Sam Chamberlain, Ph.D.
US Army Research Laboratory
Sam.Chamberlain@us.army.mil
chambesc@js.pentagon.mil
wildman@arl.army.mil
(410) 278-8948 // DSN 298
http://www.arl.army.mil/~wildman

* Current Sponsor: Joint Staff J-8 and USD(P&R) Global Force Management Community of Interest (GFM-COI); Past sponsors: the DoD C4ISR Cooperative Research Program (www.dodccrp.org) of ASD(NII) and the Army G-8/Director of Integration.

Some References

[1] S. Chamberlain: “An Enterprise Identifier Strategy for Global Naming Across Arbitrary C4I Systems,” Proceedings of the 6th International Command & Control Research & Technology Symposium, USNA, Annapolis, MD; 19 - 21 June 2001 http://www.dodccrp.org/events/zip_folders/6th_ICCRTS.zip

[2] *Enterprise Identifiers For Logistics - An Approach in Support of Army Transformation Initiatives*, http://arch-odisc4.army.mil/Data_admn/html/docs/ArmyLogisticsStudy_xFinal.pdf.

[3] M. Boller, “Common Understanding for Transformation Brigades,” *Military Review*, Sep-Oct 2000. http://www-cgsc.army.mil/milrev/English/SepOct00/boller.htm.

[4] T. Johnston: “Primary Key Reengineering Projects: The Problem;” *DM Review*, February, 2000, http://www.dmreview.com/master.cfm?NavID=55&EdID=1866.

[5] T. Johnston: “Primary Key Reengineering Projects: The Solution;” *DM Review*, March, 2000, http://www.dmreview.com/master.cfm?NavID=55&EdID=2004.

[6] T. Johnston: “De-Embedding Foreign Keys,” *DM Direct* (online):
Part 1, June 2, 2000. http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2308.
Part 2, June 9, 2000. http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2322.
Part 3, June 16, 2000. http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2331.
Part 4, June 23, 2000. http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2341.

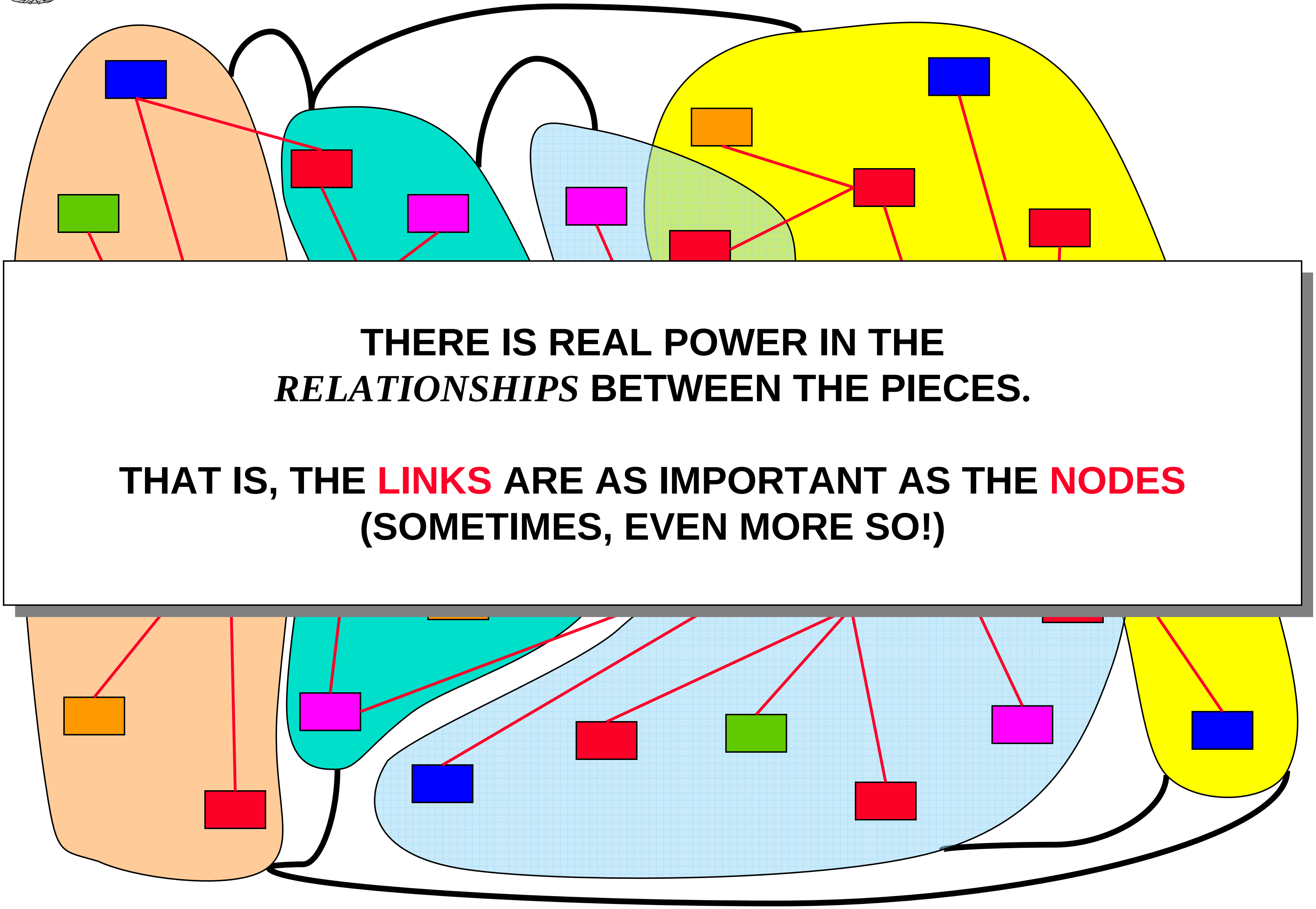
[7] S. Chamberlain: “Default Operational Representations of Military Organizations,” Army Research Laboratory Technical Report: ARL-TR-2172; February 2000. http://www.arl.army.mil/~wildman/PAPERS/tr2172.html.

[8] M. Lonigro, Mike: “The Case for the Surrogate Key”, *Intelligent Enterprise Database Programming and Design On-line*, May 1998. http//:www.dbpd.com/vault/9805xtra.htm.

[9] S. Chamberlain: “Technical Foundations for Emerging Battlefield Information Management”, Proceedings of the 1998 Command and Control Research and Technology Symposium; Naval Postgraduate School, Monterey, CA; 29 Jun - 1 Jul 98; pp 487-495. http://www.arl.army.mil/~wildman/PAPERS/jdl_98.html

[10] S. Chamberlain: “Model-Based Battle Command: A Paradigm Whose Time Has Come,” Proceedings of the First International Symposium on Command and Control Research and Technology; National Defense University, Washington, DC, 19-22 Jun 95; pp 31-38. http://www.arl.army.mil/~wildman/PAPERS/jdl_95.html

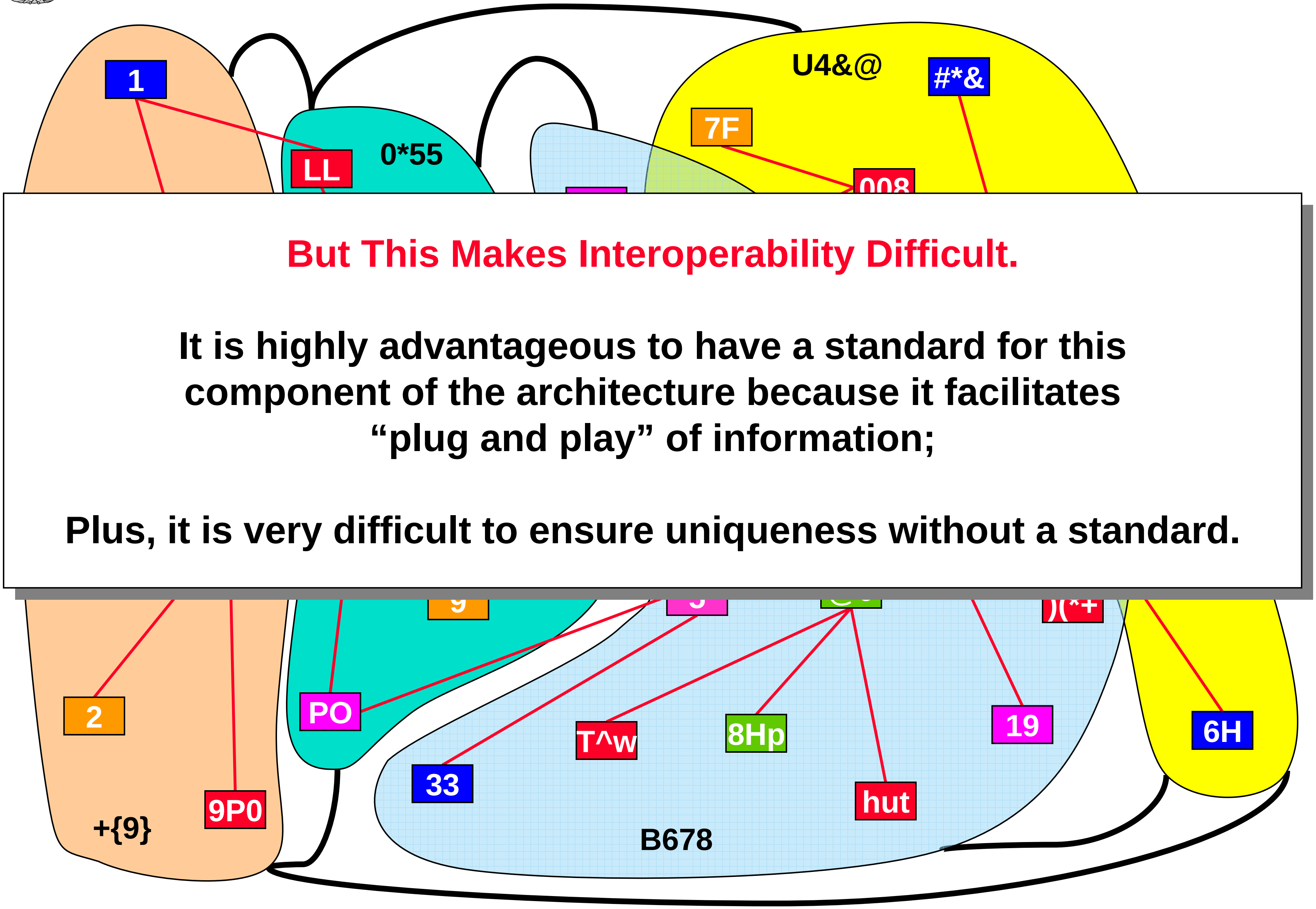
Establishing Relationships Between Data Items



THERE IS REAL POWER IN THE *RELATIONSHIPS* BETWEEN THE PIECES.

THAT IS, THE LINKS ARE AS IMPORTANT AS THE NODES (SOMETIMES, EVEN MORE SO!)

Technically, Names Must Only Be Unique, Not Consistent



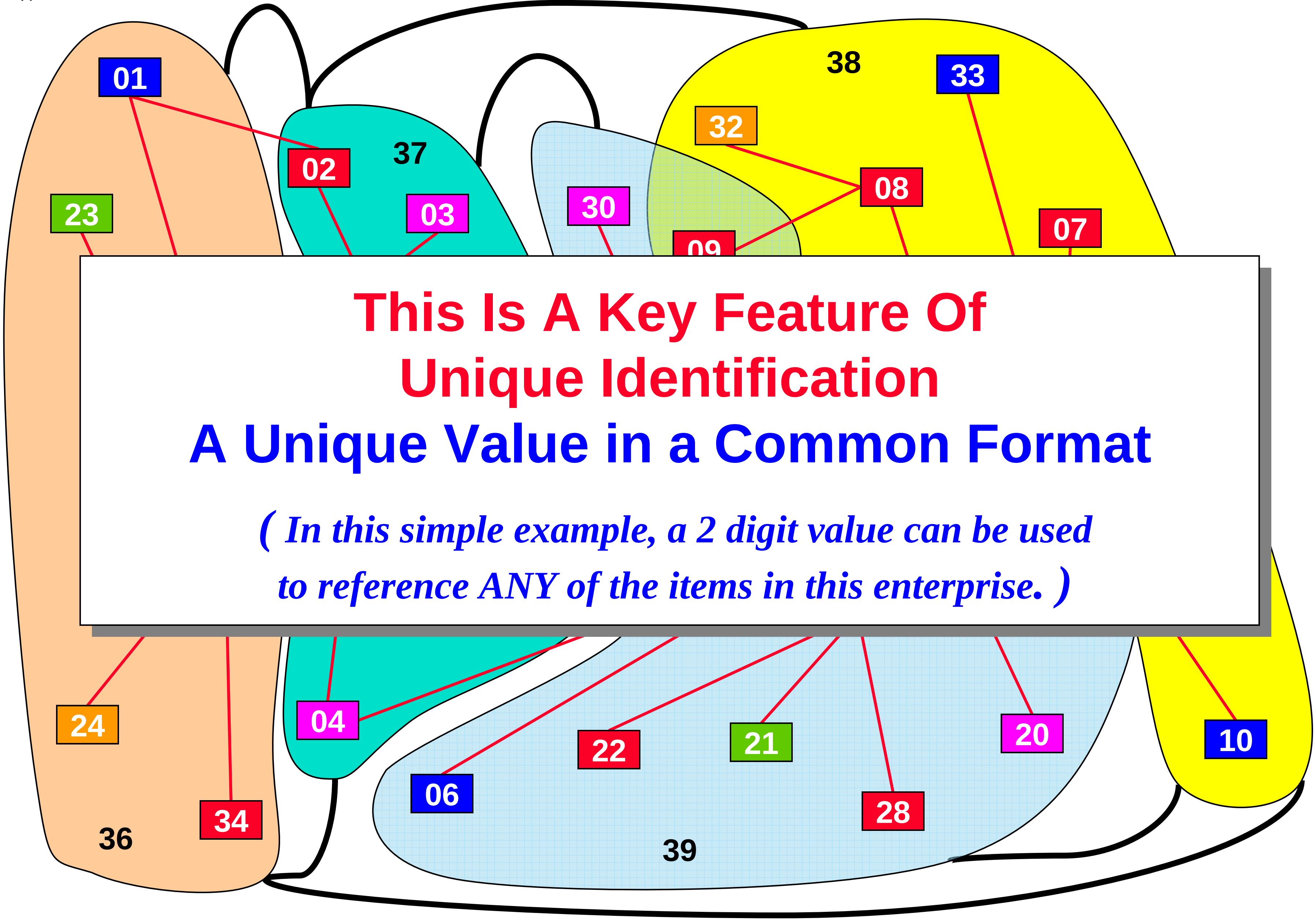
But This Makes Interoperability Difficult.

It is highly advantageous to have a standard for this component of the architecture because it facilitates “plug and play” of information;

Plus, it is very difficult to ensure uniqueness without a standard.

UNCLASSIFIED

A Common Naming Convention Across The Enterprise Is Highly Advantageous



This Is A Key Feature Of Unique Identification

A Unique Value in a Common Format

(In this simple example, a 2 digit value can be used to reference ANY of the items in this enterprise.)

Some (Relational) Database Concepts

Everything is stored in <u>tables</u> with names composed of <u>rows</u> { 1, 2, 3 } of data defined via <u>columns</u> { ID, Name, Gender, Age, Height } of attributes

Table Name: People_I_Know

ID	Name	Gender	Age	Height
1	Smith	F	26	5.1
2	Smith	M	29	5.9
3	Smith	F	47	5.3

Rows

Columns

Basic Requirement:

There must be some way to identify (or select) a single row within any table – this is the <u>Primary Key.</u>

Primary Keys (PK) with Intelligence Built In

TABLE: Pets

Pet-ID	Name	Owner
Dog-1	Fifi	Dog-2
Dog-2	Rover	Dog-2
Dog-3	Fifi	DC
Cat-1	Buck	DC
Cat-2	Tom	Cat-1
Cat-3	Tom	Cat-1a

TABLE: Pet Owners

Owner-ID	Name	Gender
Dog-2	Smith	M
Cat-1	Jones	M
DC	Brown	F
Cat-1a	Smith	F

Primary Keys (PK) w/o Intelligence
Surrogate Keys (SK)

TABLE: Pets

Pet-ID	Name	Type	Owner
1	Fifi	Dog	1
2	Rover	Dog	1
3	Fifi	Dog	2
4	Buck	Cat	2
5	Tom	Cat	3
6	Tom	Cat	4

Surrogate Key (SK)

TABLE: Pet Owners

Owner-ID	Name	Gender
1	Smith	M
2	Jones	M
3	Brown	F
4	Smith	F

Surrogate Key (SK)

E.g., “Auto-Indexing” in Microsoft Access

Enterprise Keys (EK) – Surrogate Keys Across the Enterprise

TABLE: Pets

Pet-ID	Name	Type	Owner
01	Fifi	Dog	04
02	Rover	Dog	04
03	Fifi	Dog	12
04	Buck	Cat	12
05	Tom	Cat	13
06	Tom	Cat	14

Enterprise Key (EK)

TABLE: Pet Owners

Owner-ID	Name	Gender
04	Smith	M
12	Jones	M
13	Brown	F
14	Smith	F
61	Chen	M
62	Lopez	M
63	Brown	F
64	Smith	F

Owner-ID	Name	Gender
61	Chen	M
62	Lopez	M
63	Brown	F
64	Smith	F

Review - Identifier Terms (Relational Database Perspective)

- **Primary Key (PK)** is a set of attributes in a database table that uniquely identifies an entry.

  When building relationships between entities, the primary key is often built by combining primary key fields from the associated tables plus some new fields. This produces primary keys of many different sizes and "shapes."

- **Surrogate keys (SK)** are single attribute (column) PKs with no embedded intelligence.

  One can glean nothing about the data from the key that identifies it. Therefore, they are impervious to change (there is no reason to change them).

- **Enterprise keys (EK)** are SKs that are unique across all the tables of the enterprise.
  - By definition, they are a common, single attribute (i.e., all have the same size and shape).
  - One needs a way to guarantee uniqueness.

- **Business Keys** are alternate keys, often with intelligence, for use by the database users.

- **Object IDs (OID)** are the object technology equivalent of a PK and are typically EKs inside the object database.

Example – Current Relational Scheme

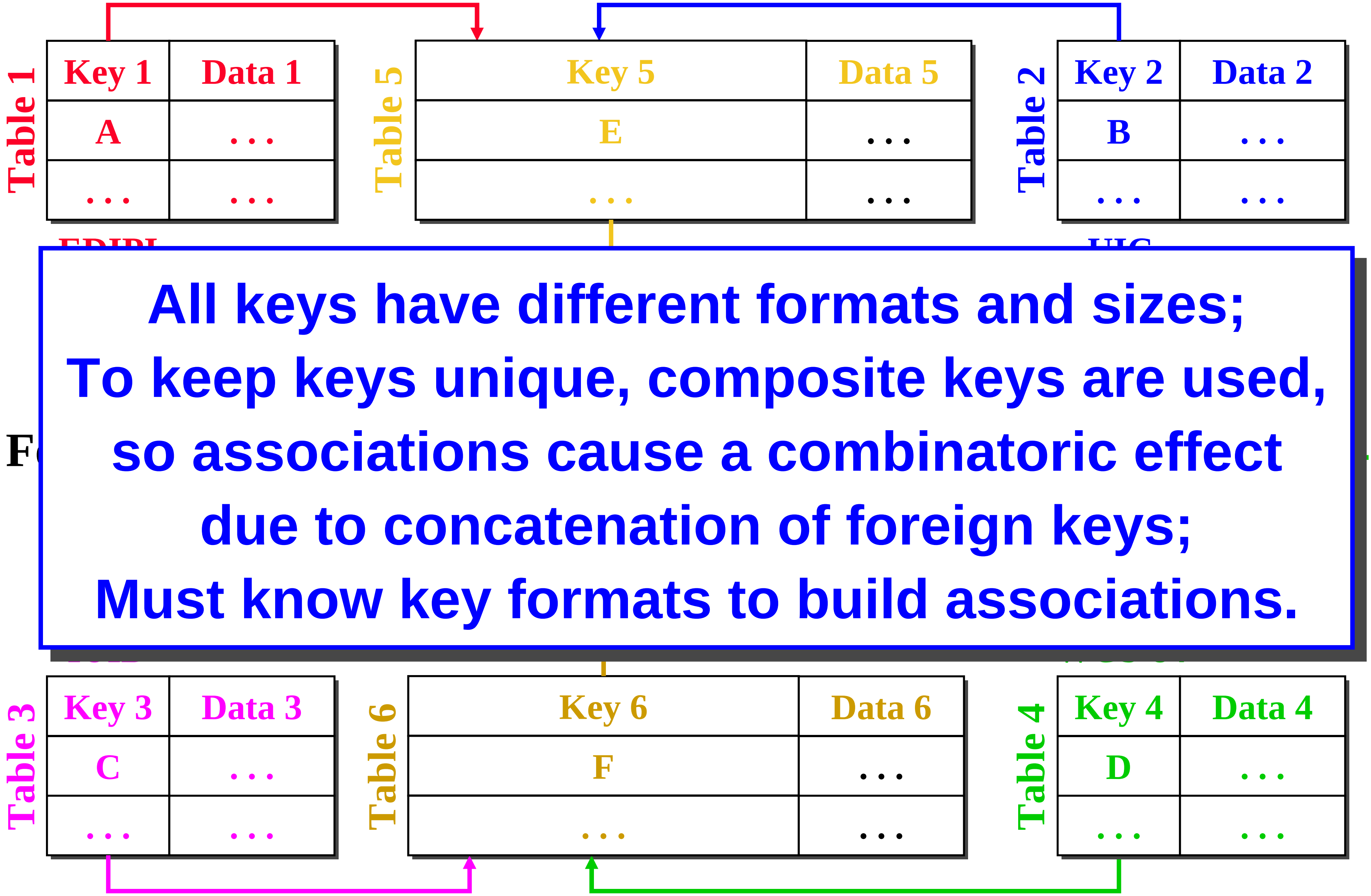


Table 1

Key 1	Data 1
A	. . .
. . .	. . .

FDIPI

Table 5

Key 5	Data 5
E	. . .
. . .	. . .

Table 2

Key 2	Data 2
B	. . .
. . .	. . .

UIC

Table 3

Key 3	Data 3
C	. . .
. . .	. . .

Table 6

Key 6	Data 6
F	. . .
. . .	. . .

Table 4

Key 4	Data 4
D	. . .
. . .	. . .

All keys have different formats and sizes;
To keep keys unique, composite keys are used,
so associations cause a combinatoric effect
due to concatenation of foreign keys;
Must know key formats to build associations.

Example – Current Relational Scheme (cont)

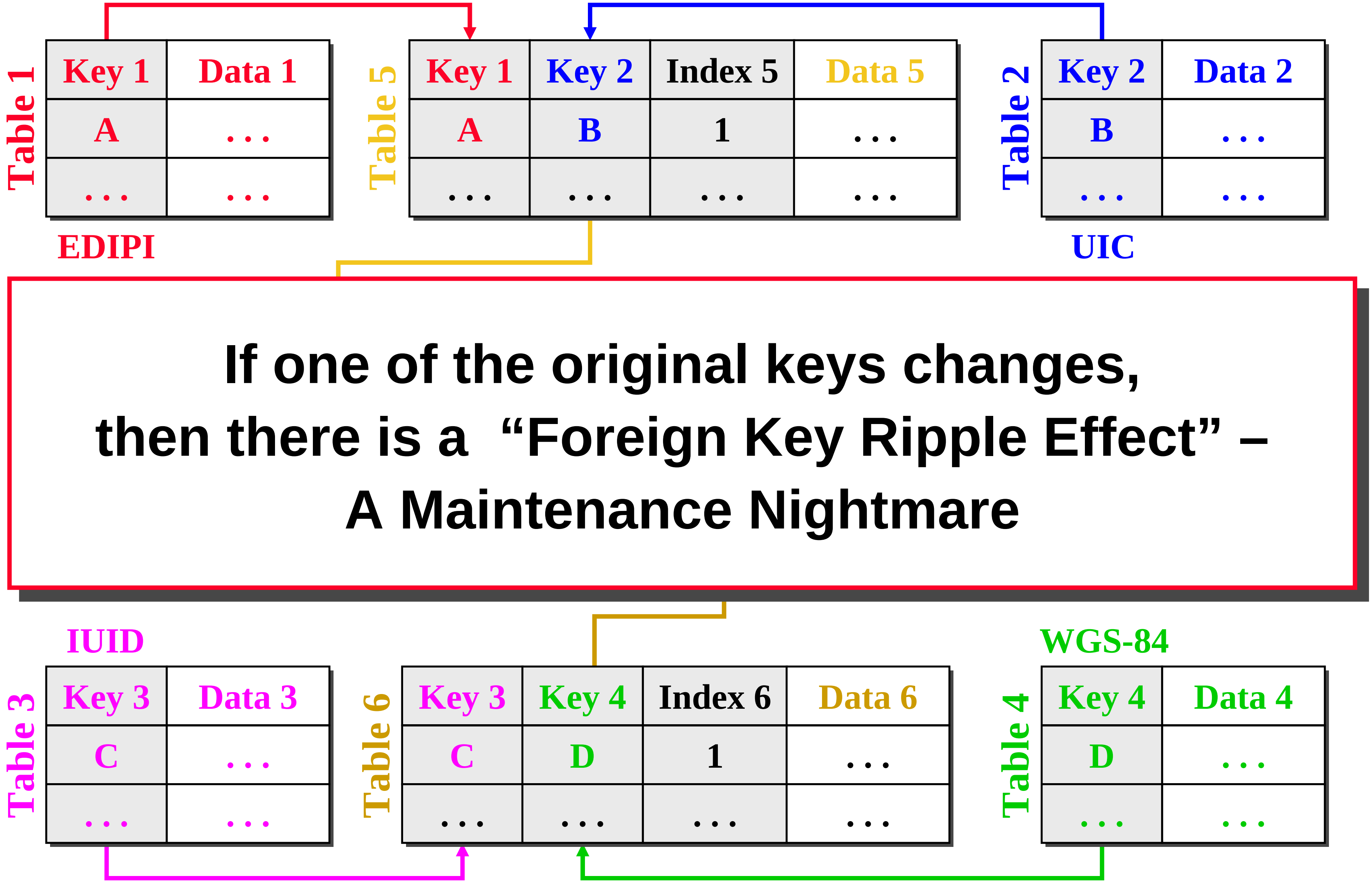


Table 1

Key 1	Data 1
A	. . .
. . .	. . .

EDIPI

Table 5

Key 1	Key 2	Index 5	Data 5
A	B	1	. . .
. . .	. . .	. . .	. . .

Table 2

Key 2	Data 2
B	. . .
. . .	. . .

UIC

If one of the original keys changes, then there is a “Foreign Key Ripple Effect” – A Maintenance Nightmare

IUID

Table 3

Key 3	Data 3
C	. . .
. . .	. . .

Table 6

Key 3	Key 4	Index 6	Data 6
C	D	1	. . .
. . .	. . .	. . .	. . .

WGS-84

Table 4

Key 4	Data 4
D	. . .
. . .	. . .

Same Example with Pure EKs

For all i,j, Format(Key i) = Format(Key j)

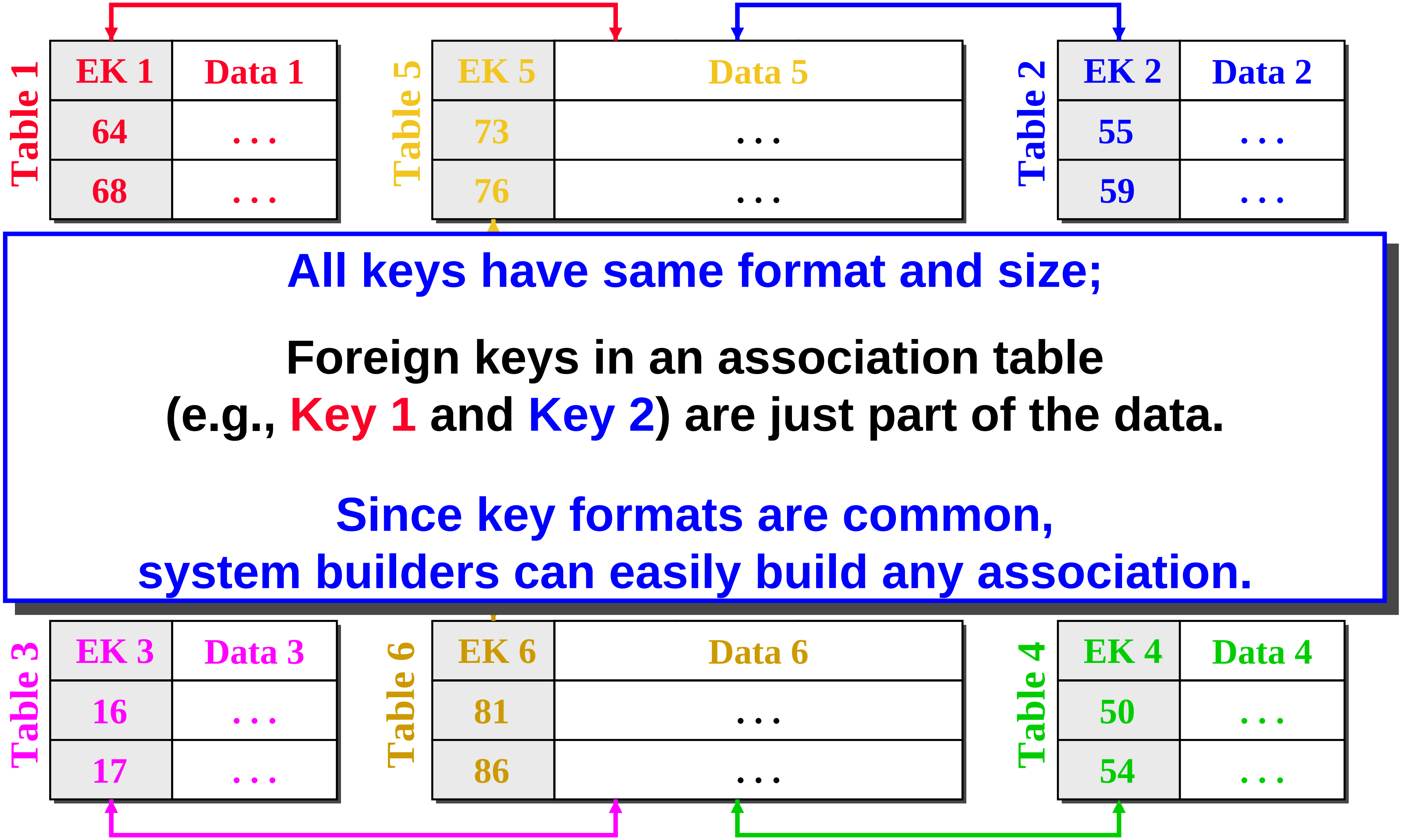


Table 1

EK 1	Data 1
64	. . .
68	. . .

Table 5

EK 5	Data 5
73	. . .
76	. . .

Table 2

EK 2	Data 2
55	. . .
59	. . .

All keys have same format and size;

Foreign keys in an association table (e.g., Key 1 and Key 2) are just part of the data.

Since key formats are common, system builders can easily build any association.

Table 3

EK 3	Data 3
16	. . .
17	. . .

Table 6

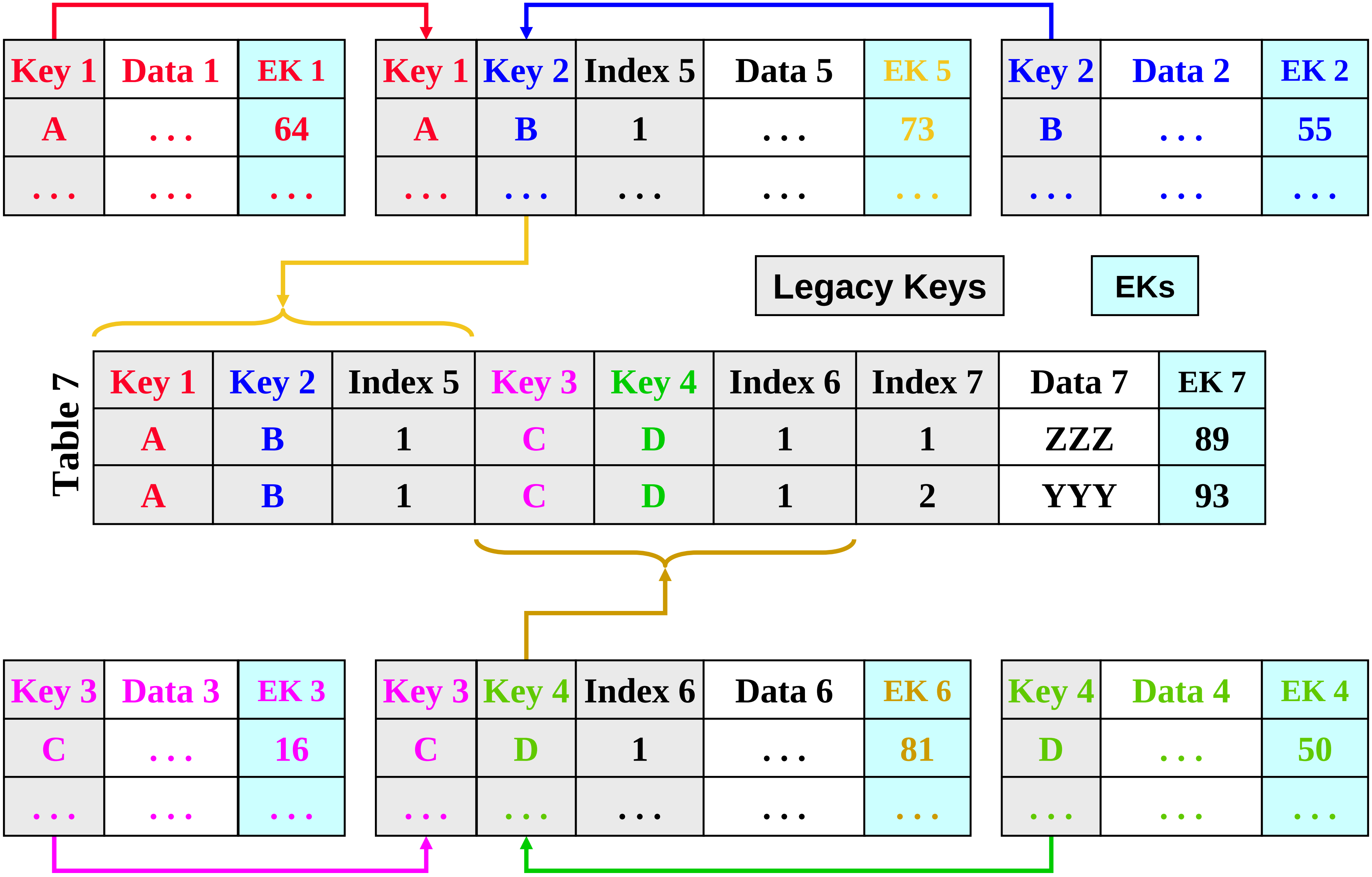
EK 6	Data 6
81	. . .
86	. . .

Table 4

EK 4	Data 4
50	. . .
54	. . .

Same Example with EK as Alternate Keys

(Just Part of the Data)



Key 1	Data 1	EK 1
A	. . .	64
. . .	. . .	. . .

Key 1	Key 2	Index 5	Data 5	EK 5
A	B	1	. . .	73
. . .	. . .	. . .	. . .	. . .

Key 2	Data 2	EK 2
B	. . .	55
. . .	. . .	. . .

Legacy Keys

EKs

Table 7

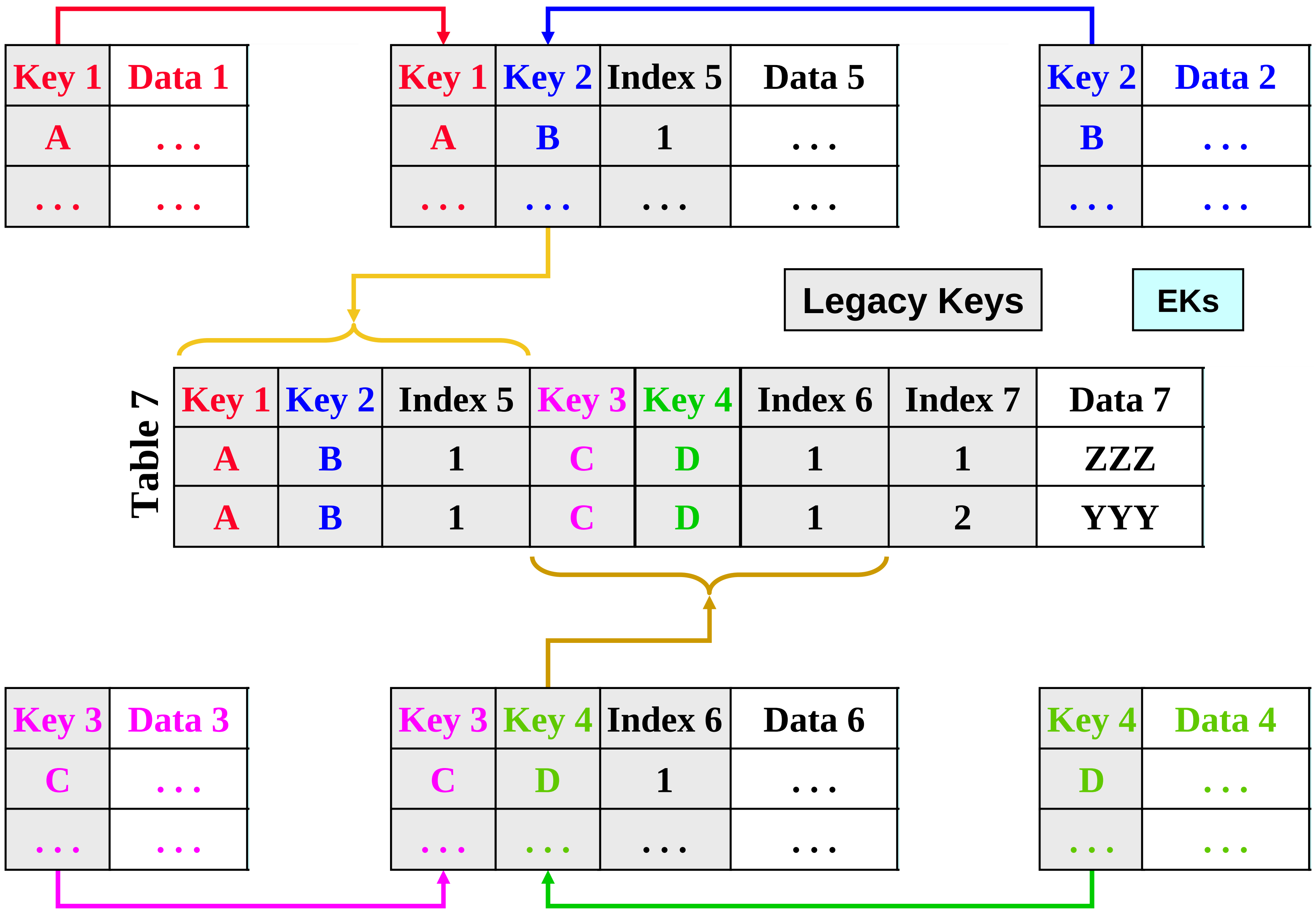
Key 1	Key 2	Index 5	Key 3	Key 4	Index 6	Index 7	Data 7	EK 7
A	B	1	C	D	1	1	ZZZ	89
A	B	1	C	D	1	2	YYY	93

Key 3	Data 3	EK 3
C	. . .	16
. . .	. . .	. . .

Key 3	Key 4	Index 6	Data 6	EK 6
C	D	1	. . .	81
. . .	. . .	. . .	. . .	. . .

Key 4	Data 4	EK 4
D	. . .	50
. . .	. . .	. . .

Key Management Compromise
Make the Index an EK so it is an Alternate Key [1]



Key 1	Data 1
A	. . .
. . .	. . .

Key 1	Key 2	Index 5	Data 5
A	B	1	. . .
. . .	. . .	. . .	. . .

Key 2	Data 2
B	. . .
. . .	. . .

Legacy Keys

EKs

Table 7

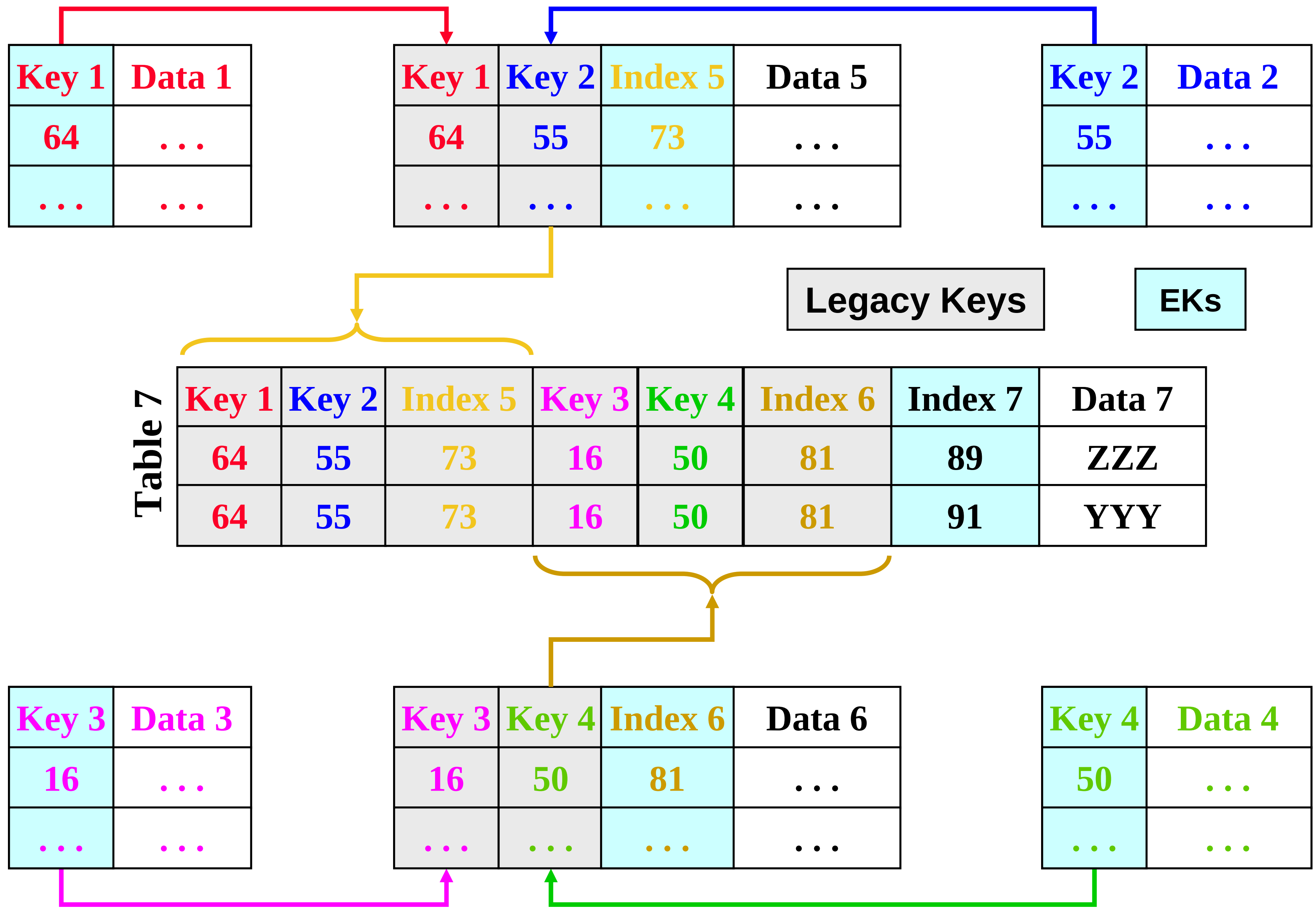
Key 1	Key 2	Index 5	Key 3	Key 4	Index 6	Index 7	Data 7
A	B	1	C	D	1	1	ZZZ
A	B	1	C	D	1	2	YYY

Key 3	Data 3
C	. . .
. . .	. . .

Key 3	Key 4	Index 6	Data 6
C	D	1	. . .
. . .	. . .	. . .	. . .

Key 4	Data 4
D	. . .
. . .	. . .

Key Management Compromise
Make the Index an EK so it is an Alternate Key [2]



Key 1	Data 1
64	. . .
. . .	. . .

Key 1	Key 2	Index 5	Data 5
64	55	73	. . .
. . .	. . .	. . .	. . .

Key 2	Data 2
55	. . .
. . .	. . .

Legacy Keys

EKs

Table 7

Key 1	Key 2	Index 5	Key 3	Key 4	Index 6	Index 7	Data 7
64	55	73	16	50	81	89	ZZZ
64	55	73	16	50	81	91	YYY

Key 3	Data 3
16	. . .
. . .	. . .

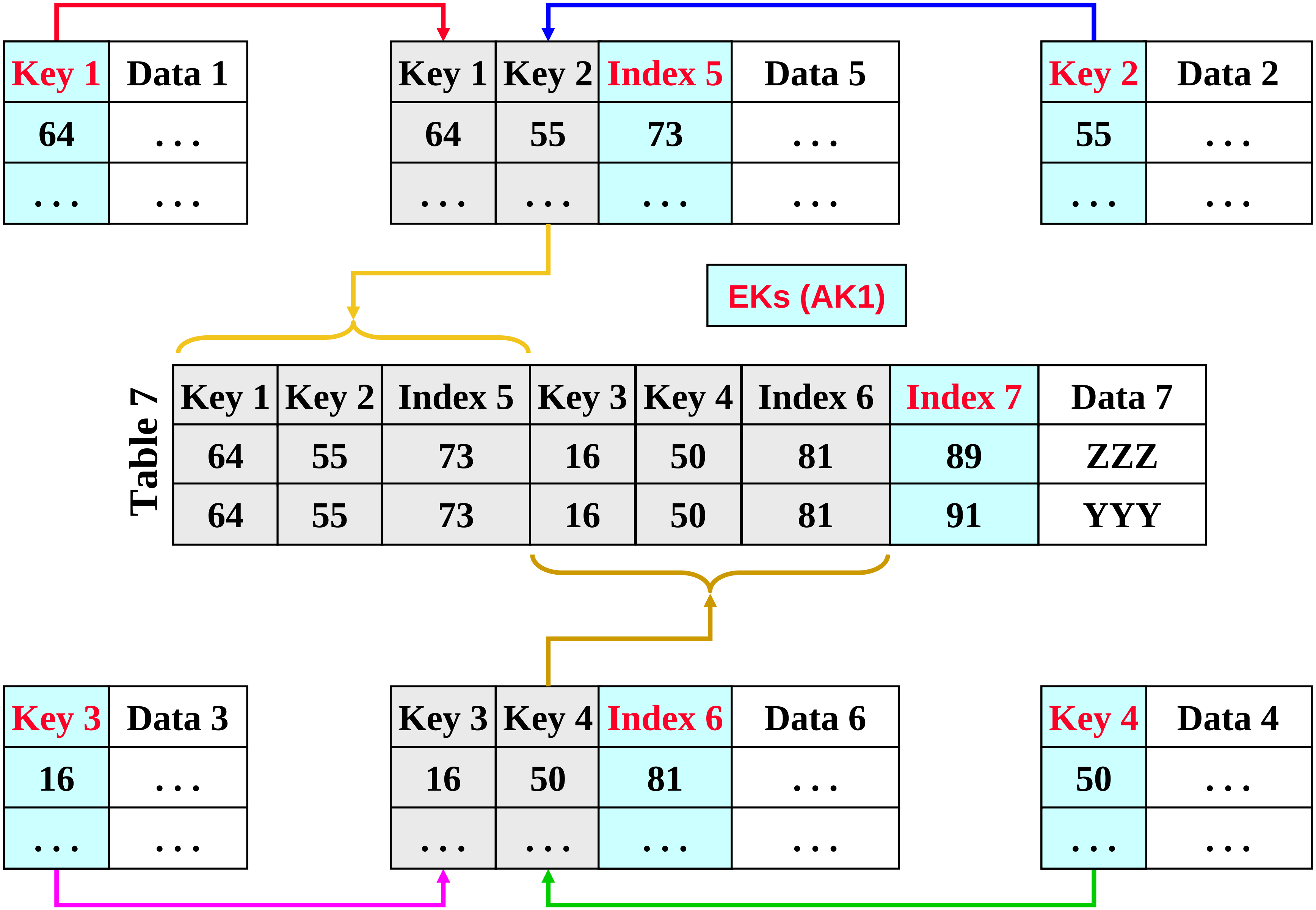
Key 3	Key 4	Index 6	Data 6
16	50	81	. . .
. . .	. . .	. . .	. . .

Key 4	Data 4
50	. . .
. . .	. . .

UNCLASSIFIED

Key Management Compromise

In the GFMIEDM IDEF1X Diagram, EKs are Annotated in Red



Key 1	Data 1
64	. . .
. . .	. . .

Key 1	Key 2	Index 5	Data 5
64	55	73	. . .
. . .	. . .	. . .	. . .

Key 2	Data 2
55	. . .
. . .	. . .

EKs (AK1)

Table 7

Key 1	Key 2	Index 5	Key 3	Key 4	Index 6	Index 7	Data 7
64	55	73	16	50	81	89	ZZZ
64	55	73	16	50	81	91	YYY

Key 3	Data 3
16	. . .
. . .	. . .

Key 3	Key 4	Index 6	Data 6
16	50	81	. . .
. . .	. . .	. . .	. . .

Key 4	Data 4
50	. . .
. . .	. . .

What is an FMIDS?

- **Force management data is exchanged as defined in the Global Force Management Information Exchange Data Model (GFMIEDM).**
- **A Force Management Identifier, or FMIDS, is a member of a set of attributes in the GFMIEDM that serve as unique identifiers for its components (i.e., entities or tables); or**
- **FMIDS is the set of attributes that are the unique identifier for the GFMIEDM components (i.e., entities or tables).**
- **The FMIDS definition is independent of the technology used to achieve uniqueness.**

FMIDS In This Example

UID :: Unique Identification:
For Table 1, the Attribute named Key 1 provides UID
For Table 2, the Attribute named Key 2 provides UID . . .
For Table 6, the Attribute named Index 6 provides UID
For Table 7, the Attribute named Index 7 provides UID

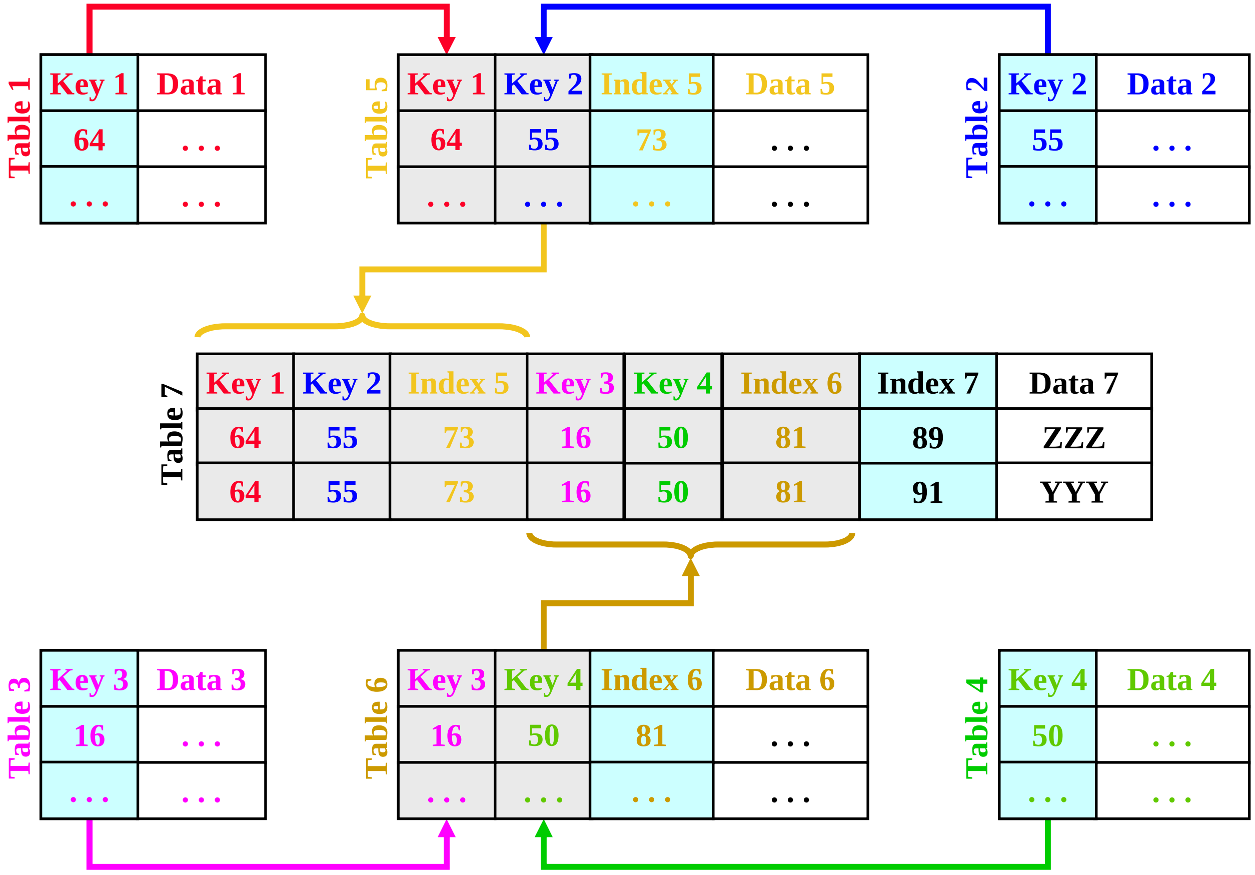


Table 1

Key 1	Data 1
64	. . .
. . .	. . .

Table 5

Key 1	Key 2	Index 5	Data 5
64	55	73	. . .
. . .	. . .	. . .	. . .

Table 2

Key 2	Data 2
55	. . .
. . .	. . .

Table 7

Key 1	Key 2	Index 5	Key 3	Key 4	Index 6	Index 7	Data 7
64	55	73	16	50	81	89	ZZZ
64	55	73	16	50	81	91	YYY

Table 3

Key 3	Data 3
16	. . .
. . .	. . .

Table 6

Key 3	Key 4	Index 6	Data 6
16	50	81	. . .
. . .	. . .	. . .	. . .

Table 4

Key 4	Data 4
50	. . .
. . .	. . .

In this example, the *FMIDS* would be:

FMIDS =

{ Key 1 (Table 1),
Key 2 (Table 2),
Key 3 (Table 3),
Key 4 (Table 4),
Index 5 (Table 5),
Index 6 (Table 6),
Index 7 (Table 7)
}

The fact that Key 1, the UID attribute for Table 1, is an attribute in Table 7 does NOT mean that it provides UID for Table 7.

Foreign Keys

To avoid confusion, it is customary not to use the same name for a key that is imported from another table – this is called a Foreign Key, marked "(FK)" in IDEF1X (e.g., Erwin) Diagrams. Below, the foreign key Key 1 in Table 5 is renamed Key A.

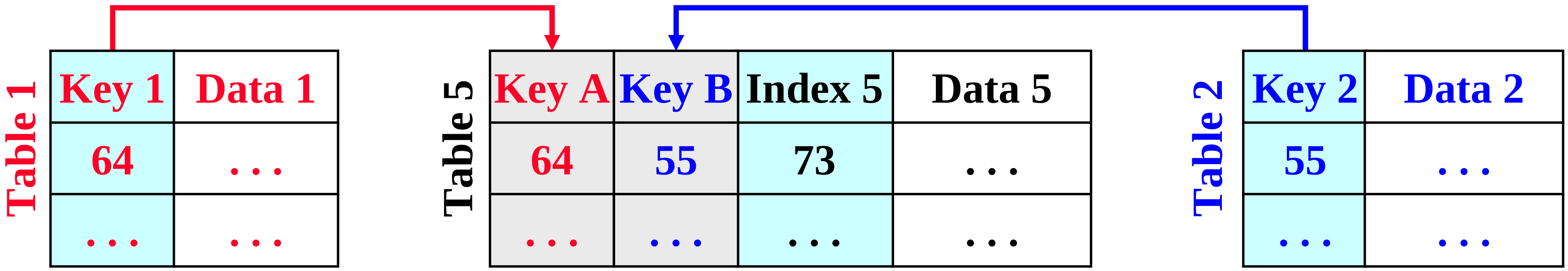


Table 1

Key 1	Data 1
64	. . .
. . .	. . .

Table 5

Key A	Key B	Index 5	Data 5
64	55	73	. . .
. . .	. . .	. . .	. . .

Table 2

Key 2	Data 2
55	. . .
. . .	. . .

In the GFMIEDM (and JC3IEDM), single attribute primary keys, like Key 1, end with the suffix "-id" (or *identifier*), while the index for multi-attribute primary keys, like Index 5, end with the suffix "-ix" (or *index)*.

These attributes are the FMIDS for their entities and are easy to identify; in the GFMIEDM Erwin diagrams, these attribute names are in red text.

Generalization Hierarchies and FMIDS

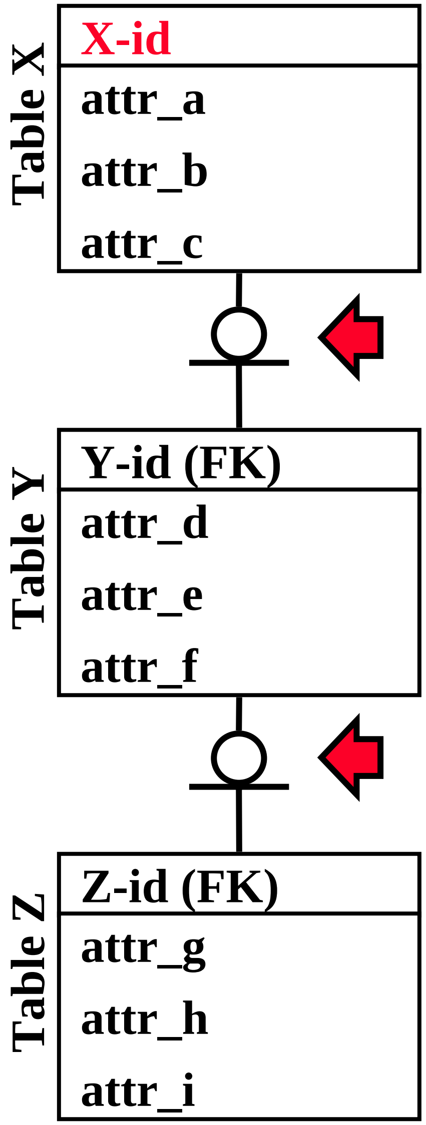


Table X

X-id

attr_a

attr_b

attr_c

Table Y

Y-id (FK)

attr_d

attr_e

attr_f

Table Z

Z-id (FK)

attr_g

attr_h

attr_i

A generalization hierarchy (GH) is a structured grouping of entities that share common attributes. It is used to represent common characteristics among entities while preserving their differences. It is the relationship between an entity and one or more refined versions. The entity being refined is called the <u>super-type</u> (e.g., Table X) and each refined version is called the <u>sub-type</u> (e.g., Table Y and Table Z).

The key of the sub-type is a foreign key from its super-type.

This means that the value of Z-id = Y-id = X-id. If X-id is an unique identifier for Table X, then Y-id is the unique identifier for Table Y and Z-id for Table Z.

A complete path of tables is always created from a sub-type to its top super-type (or vice versa, depending on ones perspective – both with the same result). Therefore, in this example, a row in Table X is never created without a row in Table Y and a row in Table Z.

In the GFMIEDM diagrams, only the top unique identifier of a GH is marked in red because the other unique identifiers are inherited at the time of creation.

Generalization Hierarchies and FMIDS

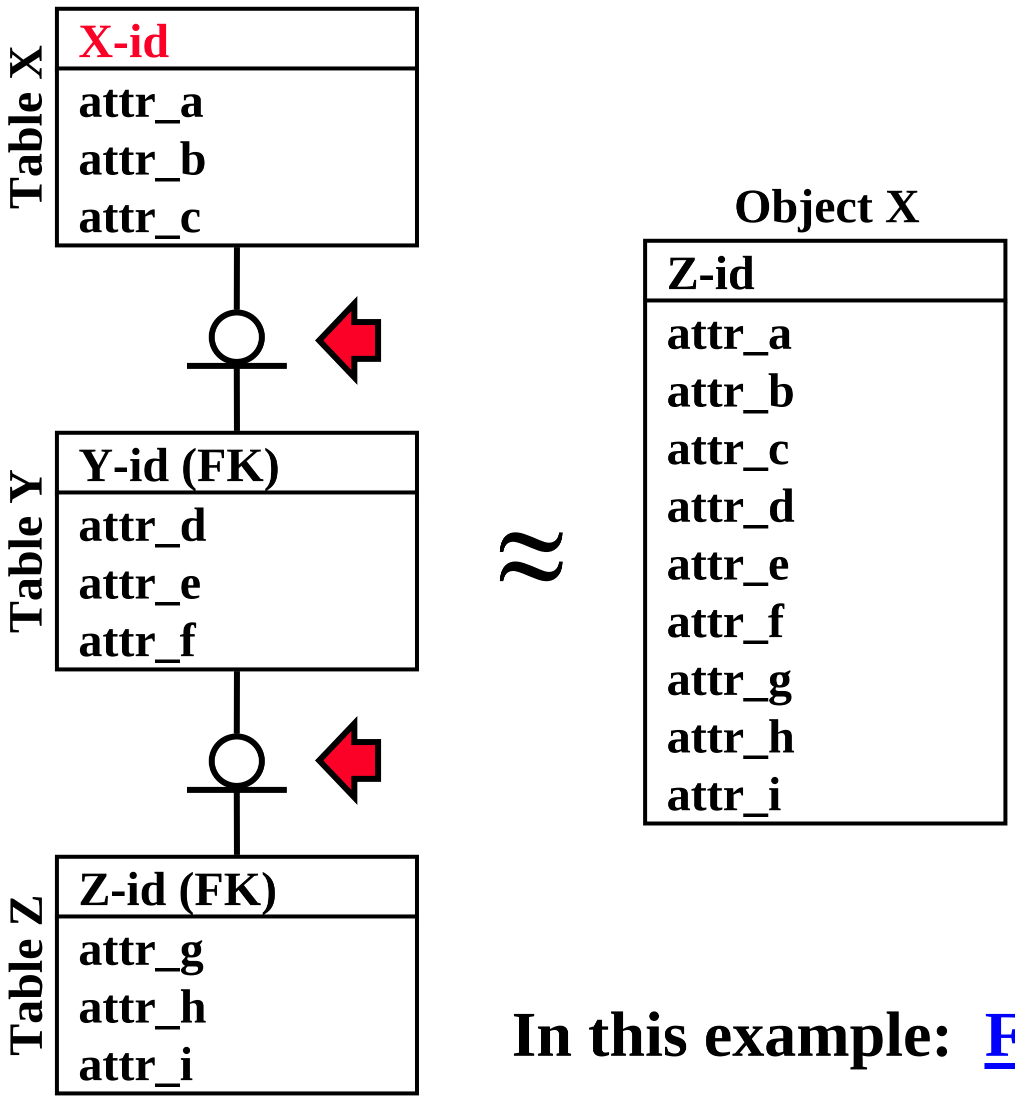


Table X
X-id
attr_a
attr_b
attr_c

Table Y
Y-id (FK)
attr_d
attr_e
attr_f

Table Z
Z-id (FK)
attr_g
attr_h
attr_i

≈

Object X
Z-id
attr_a
attr_b
attr_c
attr_d
attr_e
attr_f
attr_g
attr_h
attr_i

In this example, one can think of creating an object of Type Z that contains all the attribute up the path of its GH super-types. When this occurs, only a single unique FMIDS is created and is re-used within the tables along the path.

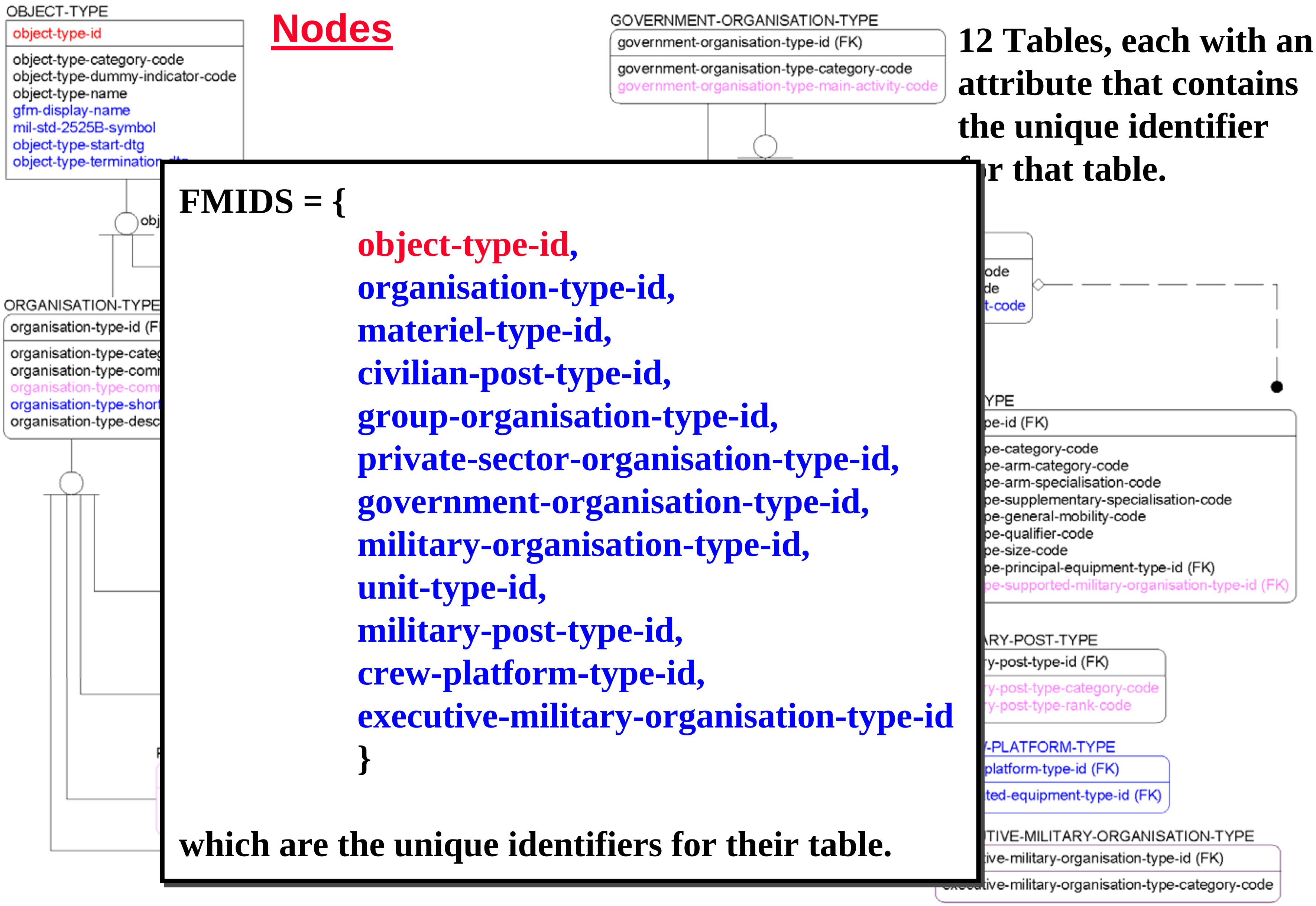
However, since several tables are used, and each table has an FMIDS attribute, ***each table's unique identifier is considered a member of the set of FMIDS for the data model*****.**

In this example: FMIDS = { X-id, Y-id, Z-id}

GFMIEDM Example 1 of FMIDS

Nodes

12 Tables, each with an attribute that contains the unique identifier for that table.


OBJECT-TYPE
object-type-id
object-type-category-code
object-type-dummy-indicator-code
object-type-name
gfm-display-name
mil-std-2525B-symbol
object-type-start-dtg
object-type-termination
obj
ORGANISATION-TYPE
organisation-type-id (F
organisation-type-categ
organisation-type-comi
organisation-type-comi
organisation-type-short
organisation-type-desc
GOVERNMENT-ORGANISATION-TYPE
government-organisation-type-id (FK)
government-organisation-type-category-code
government-organisation-type-main-activity-code
ode
de
t-code
YPE
pe-id (FK)
pe-category-code
pe-arm-category-code
pe-arm-specialisation-code
pe-supplementary-specialisation-code
pe-general-mobility-code
pe-qualifier-code
pe-size-code
pe-principal-equipment-type-id (FK)
pe-supported-military-organisation-type-id (FK)
RY-POST-TYPE
y-post-type-id (FK)
y-post-type-category-code
y-post-type-rank-code
-PLATFORM-TYPE
platform-type-id (FK)
ted-equipment-type-id (FK)
TIVE-MILITARY-ORGANISATION-TYPE
ive-military-organisation-type-id (FK)
executive-military-organisation-type-category-code

FMIDS = {

object-type-id,
organisation-type-id,
materiel-type-id,
civilian-post-type-id,
group-organisation-type-id,
private-sector-organisation-type-id,
government-organisation-type-id,
military-organisation-type-id,
unit-type-id,
military-post-type-id,
crew-platform-type-id,
executive-military-organisation-type-id
}

which are the unique identifiers for their table.

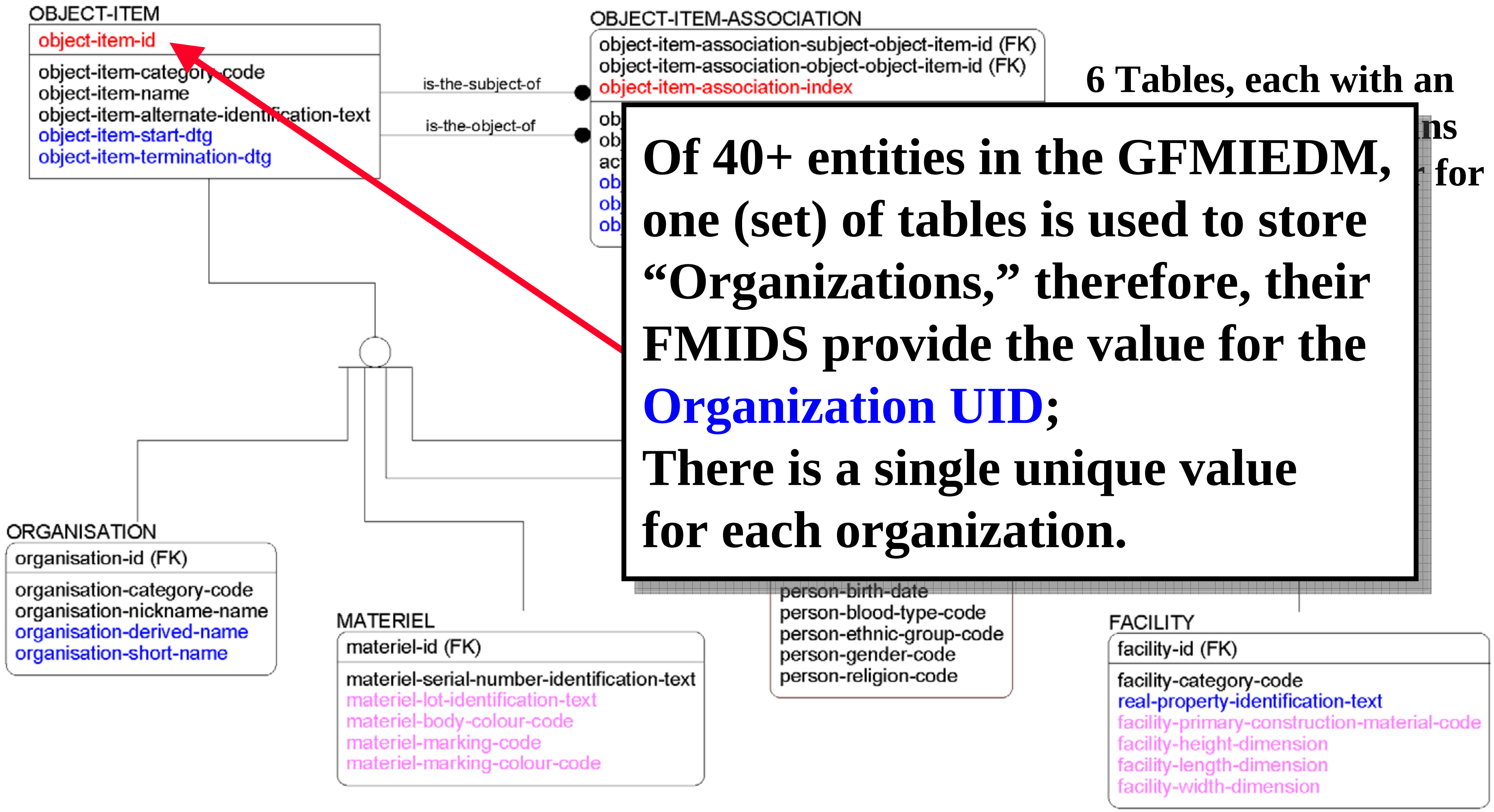
UNCLASSIFIED

GFMIEDM Example 2 of FMIDS

FMIDS = { object-item-id, organisation-id, materiel-id, person-id, facility-id, object-item-association-index } which are the EwIDs for their table.

Nodes

Links


OBJECT-ITEM
object-item-id
object-item-category-code
object-item-name
object-item-alternate-identification-text
object-item-start-dtg
object-item-termination-dtg
is-the-subject-of
is-the-object-of
OBJECT-ITEM-ASSOCIATION
object-item-association-subject-object-item-id (FK)
object-item-association-object-object-item-id (FK)
object-item-association-index
ob
ob
ac
ob
ob
ob
ORGANISATION
organisation-id (FK)
organisation-category-code
organisation-nickname-name
organisation-derived-name
organisation-short-name
MATERIEL
materiel-id (FK)
materiel-serial-number-identification-text
materiel-lot-identification-text
materiel-body-colour-code
materiel-marking-code
materiel-marking-colour-code
person-birth-date
person-blood-type-code
person-ethnic-group-code
person-gender-code
person-religion-code
FACILITY
facility-id (FK)
facility-category-code
real-property-identification-text
facility-primary-construction-material-code
facility-height-dimension
facility-length-dimension
facility-width-dimension

6 Tables, each with an ...ns ... for

Of 40+ entities in the GFMIEDM, one (set) of tables is used to store “Organizations,” therefore, their FMIDS provide the value for the Organization UID;
There is a single unique value for each organization.

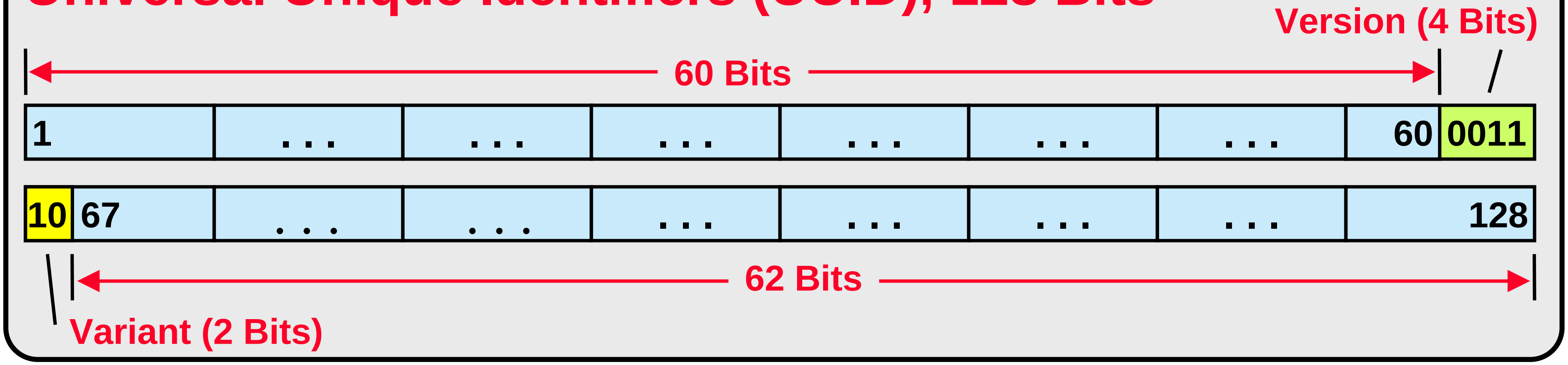
Universal Identification Technology

- **Identifier**: a property that uniquely distinguishes an item.
  [*The most basic requirement of data*.]
- **Universal Identifier**: An identifier that is unique across the enterprise (e.g., DoD).
- Fundamental Characteristics:
  - It includes no information about the entity it identifies (called a "surrogate key" in relational databases).
  - It is a fixed size (ease of software development and interoperability).
- Additional Characteristics:
  - Size
  - Allocation Scheme
- When any data is created, it is tagged with an UID that remains associated with it for its life.
- Technical Challenge: to guarantee uniqueness without sacrificing reliability and performance (i.e., no bottlenecks).

Technology for Enterprise Keys

- **Two current candidates that meet these characteristics:**

Universal Unique Identifiers (UUID), 128 Bits



Version (4 Bits)

60 Bits

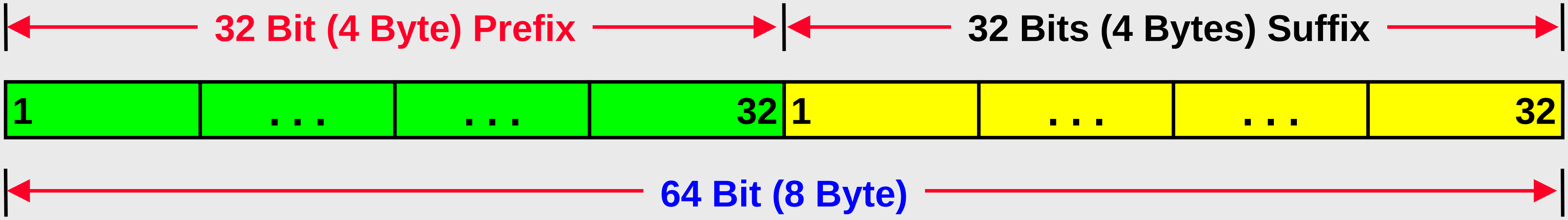
1 . . . . 60 0011

10 67 128

62 Bits

Variant (2 Bits)

Enterprise-wide Identifiers (EwID), 64 Bits



32 Bit (4 Byte) Prefix

32 Bits (4 Bytes) Suffix

1 . . . . . . 32 1 32

64 Bit (8 Byte)

Universal Unique Identifiers (UUID)

- **UUIDs were developed in the early 1980s, a 128-bit number guaranteed to be unique, for distributed processing applications.**
- **The UUID is part of ISO/IEC 11578 (1996) and costs may be incurred to implement.**
- **Five variants with mechanism that guarantees uniqueness through a combination of approaches: hardware addresses, time stamps, hash functions, and random seeds.**
- **Very common scheme for identifiers, used by many systems (e.g., MS Windows uses it to describe essentially every object in the OS). They call them Globally Unique IDs (GUIDs).**
- **There is an ongoing effort to have the specification published as a internet standard (i.e., which may be free)**

Enterprise-wide Identifiers (EwID)

- EwIDs (published as EIDs) were developed in the mid 1980s, a 64-bit number guaranteed to be unique, for distributed database applications.
- Bandwidth, simplicity, and operating system independents were key considerations
- Uniqueness accomplished via a centrally controlled portion of the identifier.
- There is an ongoing effort to have the specification published as a DoD standard (i.e., which will be free).

Three Advantages of UUID over EwIDs

1. **<u>ISO Standard</u>**

2. **<u>Allocation completely independent of any centralized control</u>**

   Built locally.

3. **<u>Incorporated in very latest programming environments</u>**

   Java 1.4., web service environments.

Three Advantages of EwIDs over UUIDs

1. **Half the size at 64-bits.**

   Not significant in commercial environments, but is in low bandwidth conditions (e.g., tactical systems); note - these systems do not use XML.

2. **Implementation (of an EwID Server) is independent of the computer's operating system (OS) and can reside completely in the applications.**

   Important for legacy systems without modern OS's, programming languages, and convenient (OS) system calls.

3. **Traceable.**

   The partially centralized allocation scheme (often considered a negative) provides an efficient technique to discover the source of an Ewid, although this should be a rare requirement.

GFM Approach: Enterprise-wide Identifiers

Concepts and Principles Behind Enterprise Identifiers (EID)

Renamed

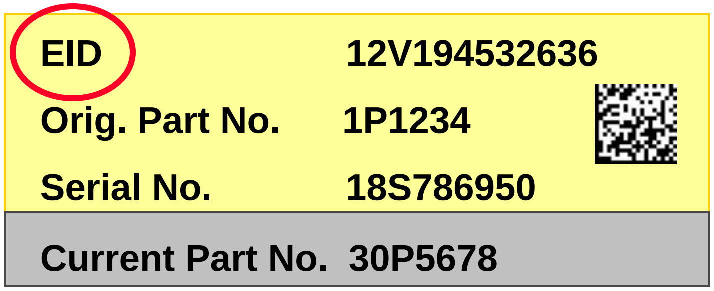
Enterprise-wide Identifiers **(EwID)**

in 2004 to avoid confusion
with the field named EID in the
USD(ATL) Tangible Item Identifier
that denotes a manufacturer
(i.e., identifies the “enterprise” that built the item)

Capture the UID Now Item UID

BUSINESS RULES

- In a database, once the UID is derived, it shall not be parsed to determine the original elements[1]



EID	12V194532636	
Orig. Part No.	1P1234	
Serial No.	18S786950	
Current Part No.	30P5678	

Record ID	UID (Constructed with a Business Rule)	EID	Orig. Part No.	Serial No.	Current Part No.	"Other Data"…
	UN1945326361234786950	194532636	1234	786950	5678	→ → →

Incremental

<u>Never Changes</u>
(mandatory for audit)

Can Change

[1] This example uses MH10.8.2 Data Identifiers.

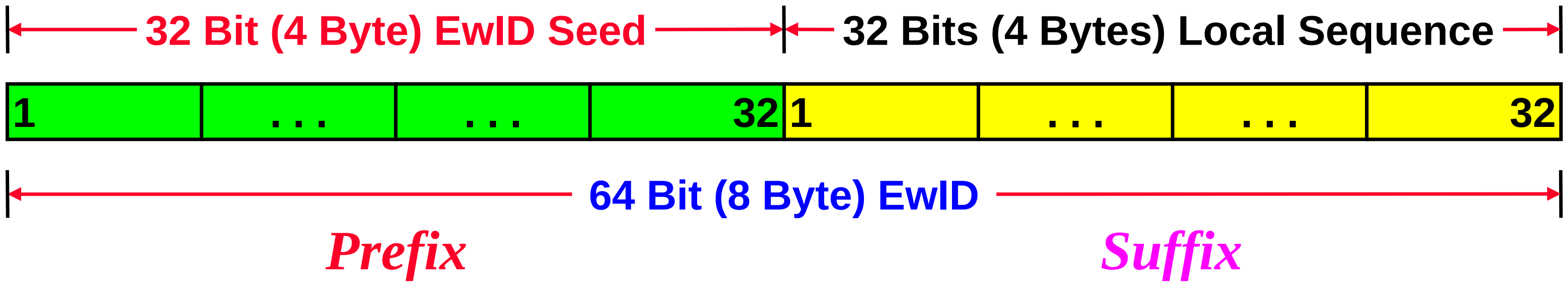
Enterprise (wide) Identifiers (EwID)

- **Identifier**: a property that uniquely distinguishes an item.
  [*The most basic requirement of data*.]
- **EwID**: An identifier that is unique across the enterprise (e.g., DoD).
- Fundamental Characteristics:
  - It includes no information about the entity it identifies (called a “surrogate key” in relational databases).
  - It is a fixed size (ease of software development and interoperability).
- Recommended Characteristics:
  - Size: ***64 bits is the smallest size that will do the job (bandwidth is a consideration)***
  - Allocation Scheme: ***Global Prefix, Local Suffix for simplicity***.
- Technical Challenge: to guarantee uniqueness without sacrificing reliability and performance (i.e., no bottlenecks).

EwID Formulation Scheme

An enterprise-wide identifier to uniquely identify *any item* in *any database* can be composed by combining unique identifiers.

First, a globally unique, four byte (32 bit) "EwID Seed" is obtained from an EwID Seed Server.



32 Bit (4 Byte) EwID Seed

32 Bits (4 Bytes) Local Sequence

1	. . .	. . .	32	1	. . .	. . .	32

64 Bit (8 Byte) EwID

Prefix

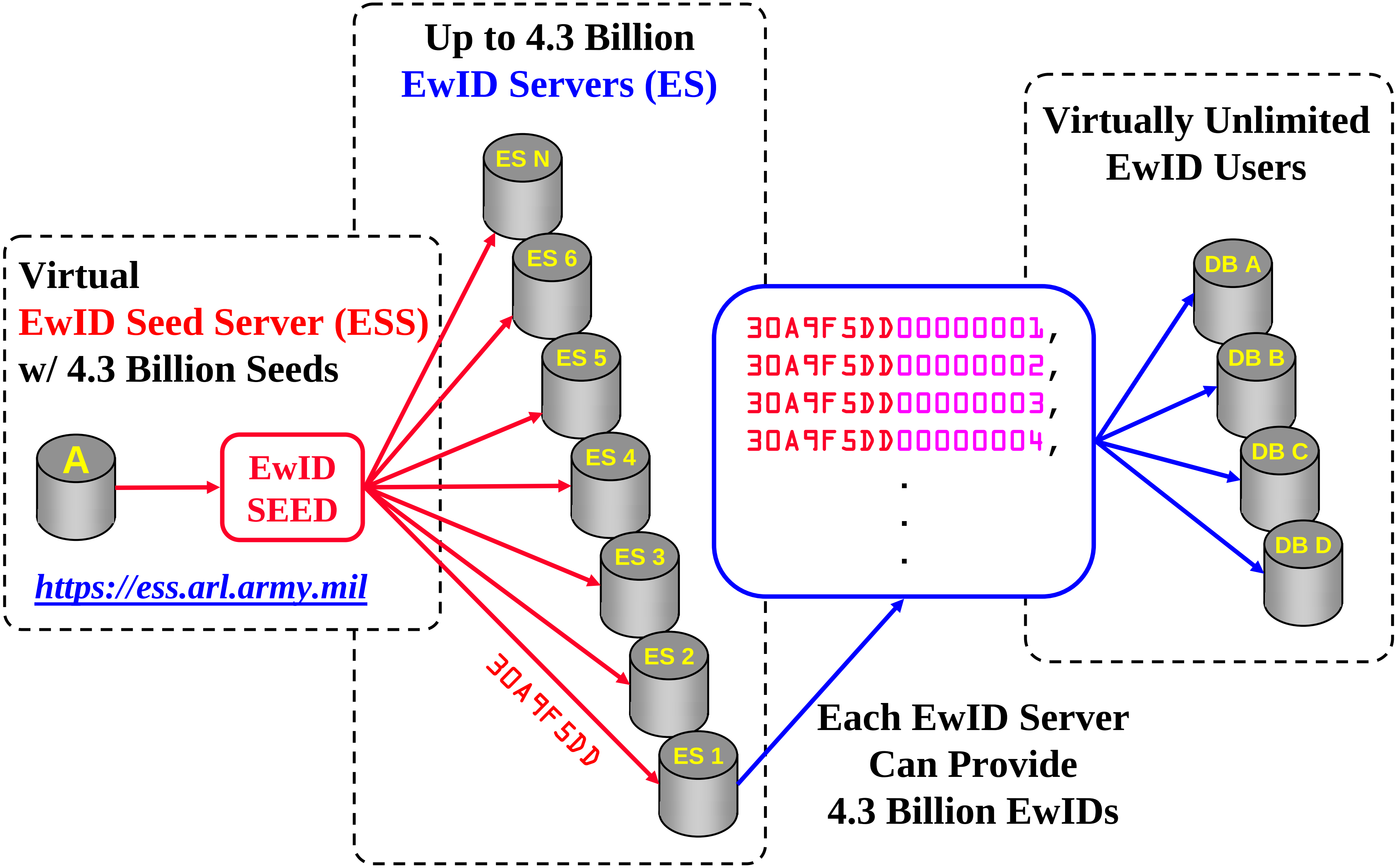
Suffix

Then, an EwID Server is established to provide EwIDs to users by producing globally unique, eight byte (64 bit) EwIDs by appending a locally controlled, unique, four byte (32 bit) suffix to the EwID Seed prefix.

The common, eight byte (64-bit) *enterprise-wide identifier* format allows
2^{64} bit patterns = 18.45 X 10^{18}, or 18.45 Exa-identifiers,
or 18 Billion Billion Unique Entities to be tracked. In other words . . .
4.3 billion EwIDs can be produced from each of the 4.3 billion EwID Seeds.

UNCLASSIFIED

Enterprise-wide Identifier Allocation Hierarchy



Virtual
EwID Seed Server (ESS)
w/ 4.3 Billion Seeds

https://ess.arl.army.mil

Up to 4.3 Billion
EwID Servers (ES)

Virtually Unlimited
EwID Users

30A9F5DD00000001,
30A9F5DD00000002,
30A9F5DD00000003,
30A9F5DD00000004,
.
.
.

Each EwID Server
Can Provide
4.3 Billion EwIDs

EwID Summary

- Data identification will have to be accomplished somehow. This is but one of many possible techniques;
  the hard part is the task of selecting one.
- Obtaining EwID Seeds is not intended to be a real-time process. This occurs when the systems are built and configured.
- EwID Seeds are free (see: *https://ess.arl.army.mil*)
- EwID characteristics & advantages:
  - ↗ No embedded information – they give away no information
  - ↗ Registration-based, this allows them to be compact & efficient (no waste)
  - ↗ Simple, fixed size – easy for software engineers to use
  - ↗ Easy to implement (add to legacy DBs as Alt Keys)
  - ↗ Data Miner’s Dream – all data is tagged with a common structure

FOR MORE INFORMATION

Dr. Sam Chamberlain
Computer & Communication Sciences Division
Computational and Information Sciences Directorate
U.S. Army Research Laboratory (ARL)

ATTN: AMSRD-ARL-CI-CT
APG, MD 21005-5067

Phone: 410-278-8948 (DSN 298)
Fax: 2934; ISDN VTC: 410-306-4620
Email: sam.chamberlain@us.army.mil or
wildman@arl.army.mil or
chambesc@js.pentagon.mil
URL: http://www.arl.army.mil/~wildman

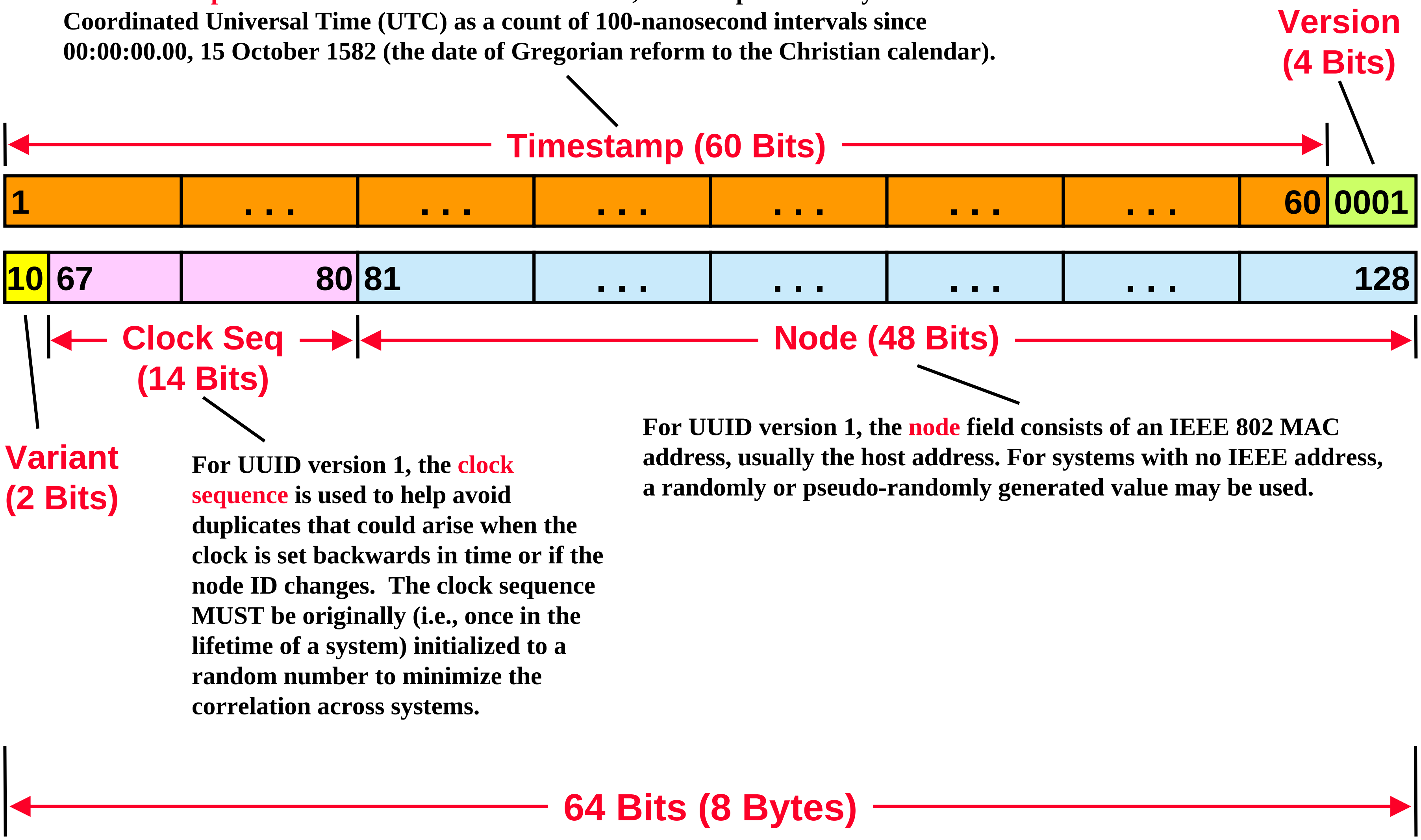
Backups

UNCLASSIFIED

Structure of a Version 1 (Time-Based) Universally Unique Identifier (UUID)

[ISO/IEC 11578:1996]

The timestamp is a 60-bit value. For UUID version 1, this is represented by Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582 (the date of Gregorian reform to the Christian calendar).

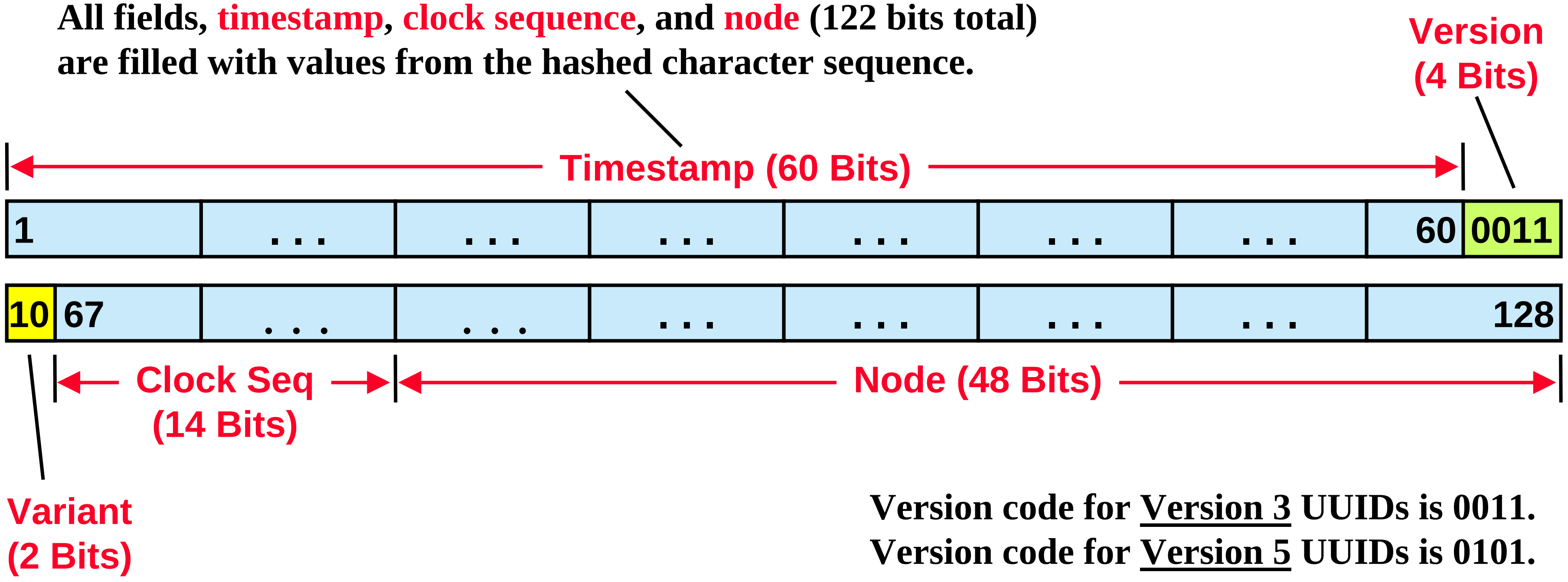


For UUID version 1, the clock sequence is used to help avoid duplicates that could arise when the clock is set backwards in time or if the node ID changes. The clock sequence MUST be originally (i.e., once in the lifetime of a system) initialized to a random number to minimize the correlation across systems.

For UUID version 1, the node field consists of an IEEE 802 MAC address, usually the host address. For systems with no IEEE address, a randomly or pseudo-randomly generated value may be used.

Structure of a Version 3 or 5 (Name-Based) UUID

All fields, timestamp, clock sequence, and node (122 bits total) are filled with values from the hashed character sequence.



Version code for Version 3 UUIDs is 0011.
Version code for Version 5 UUIDs is 0101.

Version 3 UUIDs use the MD5 (Message-Digest Algorithm #5) Hash Function by Rivest.

Version 5 UUIDs use the SHA-1 (Secure Hash Algorithm Revision 1) Hash Function by NIST.

What is a Hash Function?

From: *http://www.nist.gov/dads/HTML/hash.html* and
http://www.x5.net/faqs/crypto/q94.html

Definition: A function that maps keys (e.g., strings) to integers, usually to get an even distribution on a smaller set of values.

A hash function H is a transformation that takes a variable-size input m and returns a fixed-size string, which is called the hash value h; that is, h = H(m).

Example: h = H("The FMIDS CONOPS Page.") = 10010

Hash functions with just this property have a variety of general computational uses, but when employed in cryptography the hash functions are usually chosen to have some additional properties.

The basic requirements for a cryptographic hash function are:

- the input can be of any length,
- the output has a fixed length,
- H(m) is relatively easy to compute for any given m,
- H(m) is one-way,
- H(m) is collision-free. [By definition, to some level of probability.]

Recursive Tracking Service (Ideal Case)

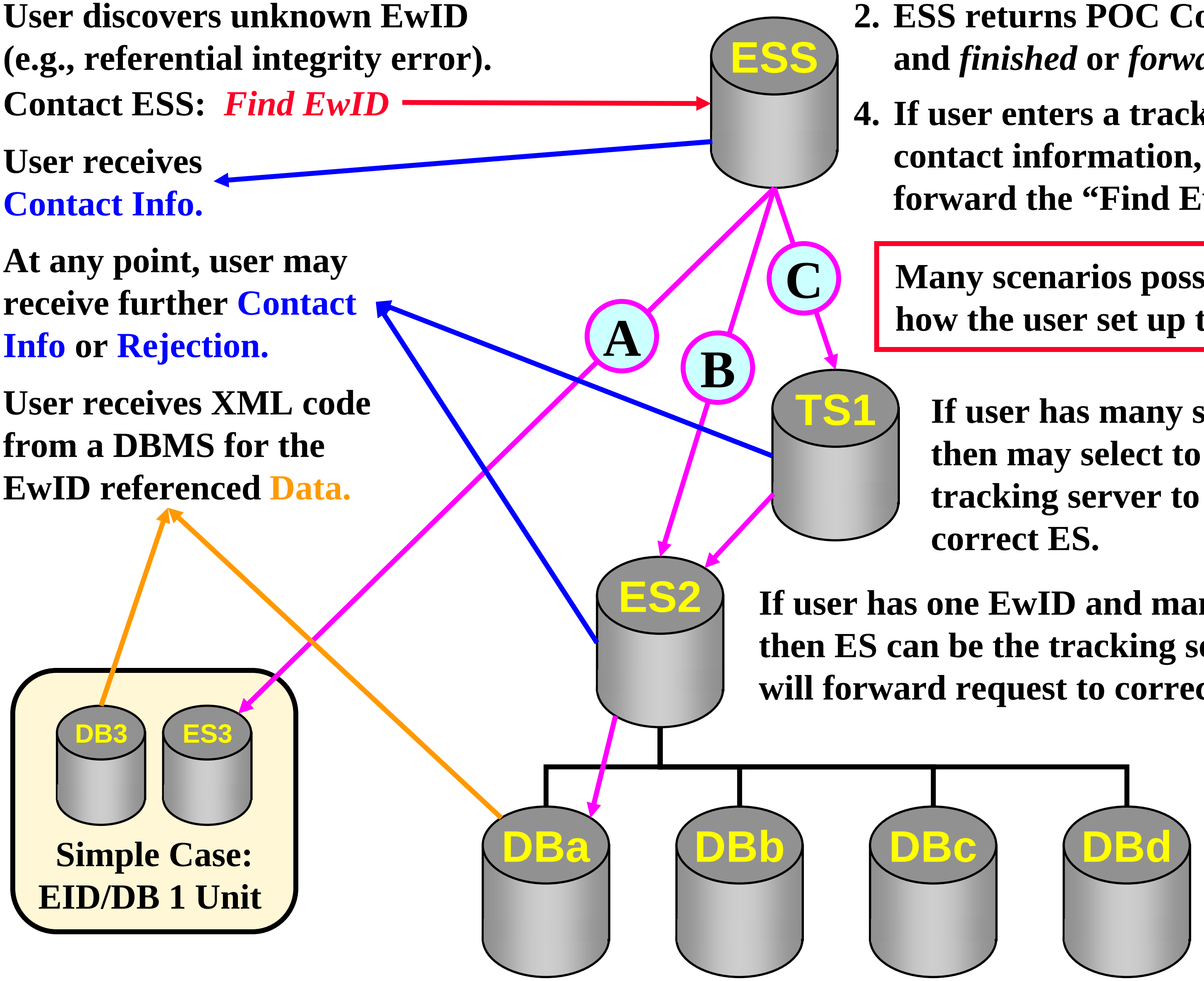
1. User discovers unknown EwID (e.g., referential integrity error). Contact ESS: *Find EwID*
3. User receives Contact Info.
5. At any point, user may receive further Contact Info or Rejection.
6. User receives XML code from a DBMS for the EwID referenced Data.

2. ESS returns POC Contac Information and *finished* or *forwarded* indication.
4. If user enters a tracking service URL in contact information, then the ESS will forward the "Find EwID" command.

Many scenarios possible depending on how the user set up the tracking service

If user has many seeds and many ESs, then may select to implement a central tracking server to forward requests to correct ES.

If user has one EwID and many DBMSs, then ES can be the tracking server and will forward request to correct database



Simple Case:
EID/DB 1 Unit